

Architectures for Vehicle Power Management

A. Harding, A. West and R.T. Watkin
Roke Manor Research Ltd, Romsey, Hampshire, SO51 0ZN

Abstract

This paper reports on work undertaken within work package 4 of the SEAS DTC project PPEM016 'Intelligent Power Management for Autonomous Vehicles' [1]. Possible architectures for an Intelligent Power Management System are discussed; in particular, we explore the interaction between power management and mission-task planning onboard an autonomous vehicle. We describe a number of potential architectures, of which two are recommended for further consideration in future work on an Intelligent Power Management demonstrator.

Keywords: Vehicle power management, distributed architectures

Introduction

Power management within an autonomous vehicle platform concerns the '*need to organise... how and when to deliver power and to which (on-board) system, whilst at the same time trying to maintain its own integrity for current and planned operations*' [2]. There are several motivators for intelligent power management in vehicle assets:

- The desire for increased efficiency in mission execution, with a consequent reduction in the cost of logistical support and over-provisioning of assets
- Increased longevity
- An anticipated increase in system complexity: increasing numbers of sub-systems and components drawing on multiple power resources
- Greater intelligence in power distribution and resource scheduling

It is thus desirable that autonomous vehicles make efficient, managed, use of available power sources. Therefore we assert that an autonomous system should be equipped with a power management component which encapsulates and controls the power characteristics of the system.

The aims of the overall PPEM016 programme of work are: to develop a generic and scalable Intelligent Power Management (IPM) outline architecture, to capture generic system level requirements for an IPM with corresponding performance metrics and to establish a quantitative assessment framework. We present the output of our architecture study component of PPEM016, the aim of which is to investigate the possible architectures for an Intelligent Power Management System, in order to support the development of a generic and scalable IPM architecture solution as part of a future demonstrator programme.

Assumptions and Requirements

A number of assumptions and requirements guided our study.

Firstly, our study considers architectures for goal and power management planning for and within a single uninhabited autonomous vehicle. We use exemplar vignettes to bound the problem of determining the scope and scale of assets onboard a UXV; vignettes are assumed to be undertaken by one autonomous vehicle carrying a limited payload of power source, stores, and consumers. We do not exclude regenerative power systems.

Secondly, we intend that an architecture developed by this work be deployed in a future demonstrator programme. We do not at this stage prescribe the precise nature of the demonstration domain, nor the demonstrator vehicle and its physical composition. We attempt to keep our architectural considerations and reasoning as generic as possible, within own-vehicle power management. In particular, we do not at this stage mandate implementation and algorithmic details for the demonstrator, and we assume that the demonstrator will take advantage of existing vehicle management systems, in particular with regard to control over the moving parts of the vehicle.

Finally, we intend that our architecture is scalable. We use the term ‘asset’ to mean a self-contained tangible component of an individual autonomous vehicle, such as a petrol engine or a video camera. Assets may be clustered into subsystems. We wish to provide the capability to change the number and types of assets by modifying the software configuration at runtime, rather than at development time. We wish further to have the flexibility to host subsystems on different physical platforms (both within the vehicle and split between the vehicle and off-board processors e.g. at a control station) and to modify that configuration at runtime. However, the total number of assets will not be large due to constraints of vehicle payload, power source capacity and the desire for the demonstrator to not become overly complex.

Architectural Components

A review of relevant background material has led to the identification of a number of functional entities that are required of intelligent power management architectures. We describe the functionality required and the design options for their implementation. Fundamentally, the design options hinge on

the degree of distribution in the system, as will be shown.

Power Management – there are two parts to this functionality: monitoring the power resources and requirements of an asset, and controlling the behaviour of the asset. Power management may be implemented in a centralised or distributed fashion.

Planning – this provides the planning functionality that considers the current situation, power constraints, and the goals to be achieved, and develops a detailed plan for how (if possible) to achieve the goals. We consider two design options:

1. A monolithic approach using a single centralised program to handle all of the assets in the system, which encapsulates the required planning and power management intelligence. All the information needed to compute a plan is made available to the planner
2. A distributed agent-based approach consisting of multiple distributed agents for monitoring and control of the assets. Figure 1 depicts the options for distributing the planner functionality. This approach consists of a number of planners distributed throughout the system.

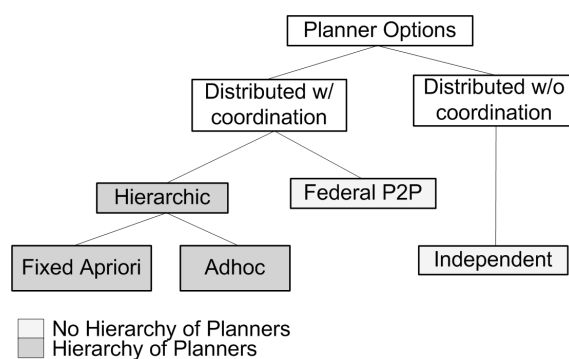


Figure 1: Distributed planner options

Planners are associated (logically or physically) with constituent subsystems of the overall system. Within a subsystem, a local plan is made by the appropriate planner based on the information that is available in that sub-system. The planner

will not generally have access to system-wide information and it is recognised that the plan devised may conflict with a plan made independently for another part of the system. Therefore, overall coordination functionality is desirable to ensure consistency and to resolve conflicts. This plan coordination function could be a single coordinator, a distributed coordinator, or a human operator.

Alternatively, it is conceivable to dispose of plan coordination, meaning that no overall plan is computed. As a result the planners could make conflicting plans. This approach may also lead to emergent behaviours, and the system simply ‘drifts’ into performing the appropriate – or more likely inappropriate – behaviour with little or no communication between the system entities.

Communication between the planner and the power management systems – this functionality is identified as being critically important for an efficient and intelligent power management system. There are two aspects to this functionality: the communication mechanism itself (over and above standard communications protocols), and the communication language.

We propose that blackboard architectures [3,4] form a basis for the communication mechanism between the power management domain and the mission planning domain. The blackboard facilitates multi-agent communications by providing a central store for information that can be written to and read by the agents. Note there are alternatives to blackboard architectures, for example the monolithic approach outlined above, but blackboards are considered to be eminently suitable to problems involving large amounts of dynamic data, over which a number of problem solvers must reason.

The blackboard architecture itself may feature a single centralised blackboard, holding the system-wide world model

which is notionally available to all agents, or distributed into local blackboards holding subsystem-level information. In both cases, the blackboard is an appropriate place to implement world model consistency enforcement, for example by using truth maintenance systems [5,6].

The information required by the planner is semantically different from the raw data provided by the power management functionality. Thus a semantic translation function is required in the system. This function has several possible architectural arrangements: the presence or absence of an explicit translator function; and the centralisation or distribution of the function. Thus, for example, there is a difference between a centralised explicit translation function (an identifiable software entity handling all exchanges in the system) and implicit distributed translation (device-specific data handling is hidden in the implementation detail of some localised software entity). Explicit translation functions are recommended, because this separates the logic of the planner from the detail of the asset, in a way that permits subject matter expertise to reside in a clearly-defined place in the architecture.

Architectural Options

In principle, the architectural options are as follows:

1. Power management: centralised or distributed
2. Blackboard: centralised or distributed
3. Semantic translation: implicit or explicit for each of centralised and distributed, giving four options
4. Planner options: centralised, distributed with coordination, and distributed without coordination

However these options are not completely independent. Our approach was to choose an option for one component and analyse

the impact that had on the options available for the other components. For example, the monolithic power management architecture dictates that there is no distribution inherent in the system so the only option for each of the other components is the centralised option. In addition, as stated earlier, it is recommended that the translation functionality is made an explicit function in the system. This resulted in five possible architectures, discussed in the following sub-sections.

Monolithic Architecture

The monolithic architecture is for a single executable whose components are the centralised and monolithic component options. Thus, there is a single monitor / control function that handles all the assets in the system. A central translation function is used to provide the translation of information between the assets and monitor / control. The monolithic approach uses a single, centralised planner to create an overall plan, based upon information available to the application.

The implementation of this approach is complex because of the centralised functions. The functions would have to handle the information from all the assets. Every time a new asset is to be added to or removed from the system, changes would need to be made to the monitoring, control and translation functions. This approach is therefore not very scalable.

Single Planner, Central Blackboard, Central Semantic Translation

This architecture consists of a single planner, a central blackboard and a central semantic translation function, with separate monitoring and control functions for subsystems. This architecture differs from the monolithic approach in that distributed asset control is used. Separate monitoring and control functions are used for each asset or subsystem (to an appropriate granularity). This simplifies asset handling

because the monitoring and control functions only need to deal with information from the local asset. Distributed asset handling makes this architecture scalable to the addition of new assets.

This approach decouples mission planning and asset management in two ways. Firstly, the asset handling has been distributed. Secondly, the planner and the asset handling agents are no longer part of the same function. A separate function is used for each agent and this means that the separate processes can be performed at the same time.

A centralised translation function is used for this architecture, located between the assets and the respective monitoring and control functions. This means that even though the monitoring and control functionality is distributed, information has to pass through a centralised function before reaching the relevant asset.

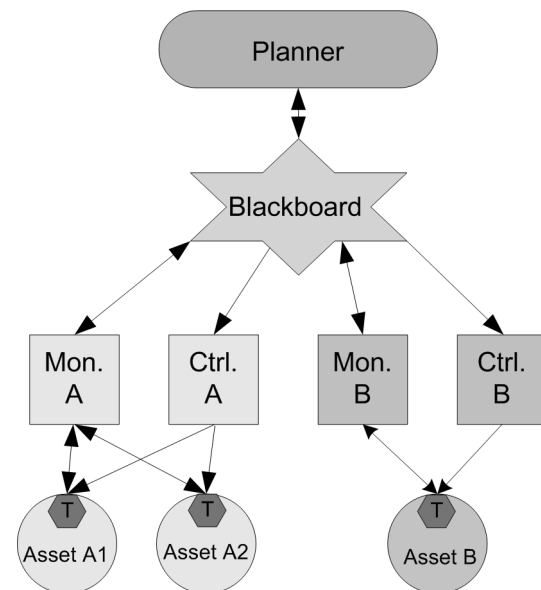


Figure 2: Centralised planning option

Single Planner, Central Blackboard, Distributed Semantic Translation

This architecture, depicted in Figure 2, is very similar to the architecture used in the Autonomous Soaring project software [7,8]. Distributed agents each handle a

different part of the system and communicate via a centralised blackboard agent. A single planner then decides on an overall mission plan based upon the knowledge available on the blackboard.

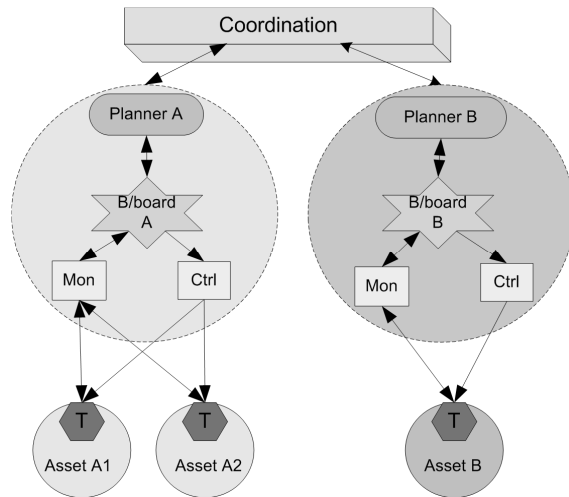


Figure 3: Distributed planning option

This approach offers the advantages that it is scalable for new assets and decouples mission planning and asset management. The difference is that the translation is now distributed throughout the system. Each asset now has its own local translation function. This addresses the counter-productivity of distributing the asset handling but then centralising the information in translation. The advantage of the distributed approach to translation is that the translation is simple and efficient because the translation function only needs to handle local information from a single asset. This approach is also more scalable than using a single translation function because if a new asset is added to the system then a new local translation function can be added to handle the new information.

Distributed Planning with Coordination

This architecture, depicted in Figure 3, provides further distribution of the system. The planners are now distributed (with coordination) and local blackboards are used.

Each asset or sub-system is handled by a local monitor and control agent. The monitoring and control agents sit within a localised group of entities that consists of a local translation function, a local blackboard and a local planning agent. Each planner makes a local plan based upon the asset information that is available to it.

The blackboards have a consistent view of their local asset or subsystem but not of the whole system. As a result the different planners could (and almost certainly will) make conflicting plans. There is therefore the need for a plan coordination function which acts as a high-level planner which oversees the local plans to ensure consistency and to resolve conflicts. This plan coordination function could consist of a single coordinator, a distributed coordinator which could make use of market-economics ideas such as contract nets [9], or a human operator who can respond when conflicting plans are produced.

The plan coordination function (in whatever form) makes use of the principles of subsidiarity and devolved autonomy by generating goals to be devolved to the appropriate subsidiary planners, by providing conflict management, and potentially by providing solution recombination.

The distributed planning architecture is scalable as assets and subsystems can be added to or removed from the system without the need to significantly update the rest of the system. A new asset can be added to a sub-system by implementing a new translation function for that asset and updating the monitoring, control and planning agents for the sub-system. The rest of the system is unaffected. Similarly, a new sub-system consisting of a blackboard, a planner, a monitoring agent and a control agent and an asset can also be added to the system. This is simpler than

having to update complex centralised functions or agents. The architecture further offers the promise of efficient system operation, because the distributed functions can run independently and concurrently per subsystem, each within an appropriately-scaled memory and processing space. Translation is simple and efficient as each translation function is only handling local information.

Distributed Planning without Coordination

This architecture is the same as described above, but without the coordination. Planning, asset control and translation are all distributed and local blackboards are used. In particular, there is no coordination between the distributed planners.

Conclusions and Future Work

Having considered the advantages and disadvantages of the five architectures described above, we identify two possible architectures for the demonstrator. They are the ‘central blackboard, distributed semantic translation’ option, and ‘distributed planning with coordination’, depicted in Figure 2 and Figure 3, respectively.

Factors that will influence the choice between these architectures include:

1. The number and complexity of assets in each subsystem;
2. The volume of data to be processed;
3. The complexity of the data to be processed;
4. The level of control required in the subsystem;
5. The amount of interaction with other subsystems; and
6. The speed and frequency of planning required for the subsystem.

We recommend that these two architectures are considered for the intelligent power management demonstrator and that following further analysis of requirements, one is chosen for the demonstrator system.

References

- [1] A. Harding, A. West, *Intelligent Power Management for Autonomous Vehicles: WP4 Architecture Study*, SEAS DTC PPEM016 Final Report, PPEM016/roke/001. (January 2009)
- [2] M. Johnson, *Concepts and Options for Power Management on Autonomous Vehicles*, SEAS DTC ADD020 Final Report ADD020-01, Draft. (March 2008)
- [3] D. Corkill, *Blackboard Systems*, in *AI Expert* 8(9) 40-47 (1991)
- [4] D. Corkill, *Collaborating Software: Blackboard and Multi-Agent Systems & the Future*, in *Proceedings of the International Lisp Conference* (October 2003)
- [5] J. de Kleer, *An Assumption-based TMS*, *Artificial Intelligence*, 28(2), 127-162. (1986) (pp. 127–162)
- [6] J. Doyle, *A Truth Maintenance System*, *Artificial Intelligence*, 12(3), 231–272. (1979)
- [7] M. Hook, D. Huish, G. Purcell, E. Sparks, R. Watkin, *Autonomous Soaring*, SEAS DTC PPEM013 Final Report, PPEM013/roke/01, Issue 1. (November 2007)
- [8] R. Watkin, C. Stennett *Autosoaring Extensions Final Report*, SEAS DTC ADD032 Final Report, ADD032/01, Issue 1, (May 2008)
- [9] R. G. Smith, *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*. In *IEEE Transactions on Computers*, C-29 N^o 12 1104-1113 (December 1980)

Acknowledgements

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.